Temporal Action Detection in Long Videos on AWS EC2

Yiyue Zhang (Team Lead), Weijing Wang, Jiyang Gao

I. Summary

The amount of videos is increasing very rapidly, as capturing videos has become much easier than before with the wide use of iPhone, GoPro and other portable devices. The need for temporal analysis of the large amount user generated videos is also increasing rapidly. For example, users may want to collect highlight moments out of the long boring videos, e.g. a fancy surfing motion, an incredible skydiving. To achieve this goal, we need a computer vision system that is able to process large scale of video data rapidly and precisely generate temporal annotations. This problem is called temporal action detection in the field of computer vision. Temporal action detection is an important problem, as fast and accurate extraction of semantically important (e.g. human actions) segments from untrimmed videos is an important step for large-scale video analysis.

In this project paper, we compare implement and train the state-of-the-art action proposal generation and action detection method, TURN [5], on AWS EC2. Based on TURN, we originally propose to use max-pooling for clip-feature modeling to capture the signature information inside the clip. Besides TURN, other state-of-the-art methods address this problem by applying action classifiers on sliding windows. Although sliding windows may contain an identifiable portion of the actions, they may not necessarily cover the entire action instance, which would lead to inferior performance.

We train different system variants in TURN to compare the C3D/Flow feature performance. We test our proposed max-pooling method in TURN and shows a performance boost. The generated proposals are applied in action detection task with SVM classifiers and SCNN detectors, and achieve state-of-the-art performance.

For computing platform and implementation, we use the codes provided by TURN [5], and train the model on AWS EC2 platform. We use a p2.xlarge instance which includes a single Nvidia K80 GPU, 4 vCPU and 61GB memory. We use the "Deep Learning Base AMI" to initialize the instance.

In the following, we first demonstrate the problem statement and methods, and then introduce the experiment settings on AWS EC2, including the environment configuration, model parameters and training details. In section 4, we will discuss the evaluation results and comparing with other methods.

II. Problem Statement and Methods

A. *Problem Statement*

Temporal action detection in long videos is an important and challenging problem, which has been receiving increasing attention recently. Given a long video, the task of action detection is to localize intervals where actions of interest take place and also predict the action categories.

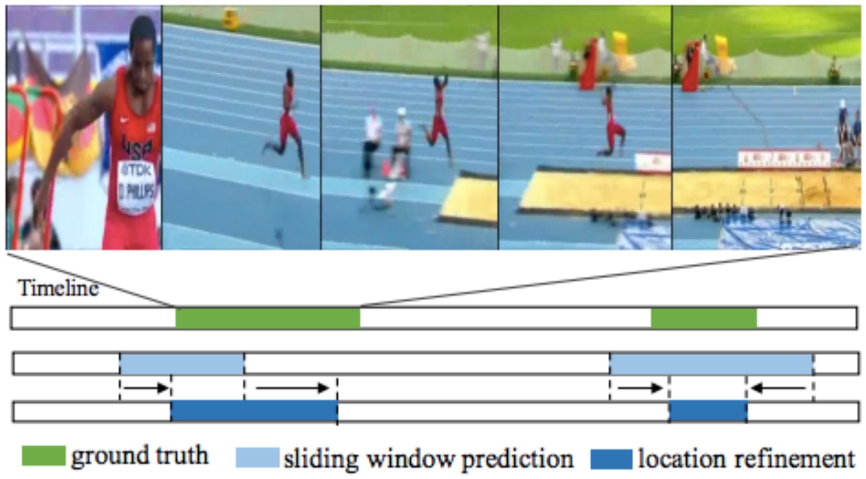

Figure 1. Temporal action detection aims at predicting the action categories and start and end time of the action.

Current state-of-the-art methods [3] on action detection extend classification methods to detection by applying action classifiers on dense sliding windows. However, while sliding windows may contain an identifiable portion of the action, they do not necessarily cover the entire action instance or they could contain extraneous background frames, which may lead to inferior performance. Similar observations have also been made for use of sliding windows in object detection [4]. Inspired by object detection, Shou et. al [3] proposed a two-stage pipeline for action detection, called SCNN. In the first stage, it produces actionness scores for multi-scale sliding windows and outputs the windows with high scores as class-agnostic temporal proposals; in the second stage, SCNN categorizes the proposals to specific actions. However, SCNN still suffers from the imprecision of sliding window intervals.

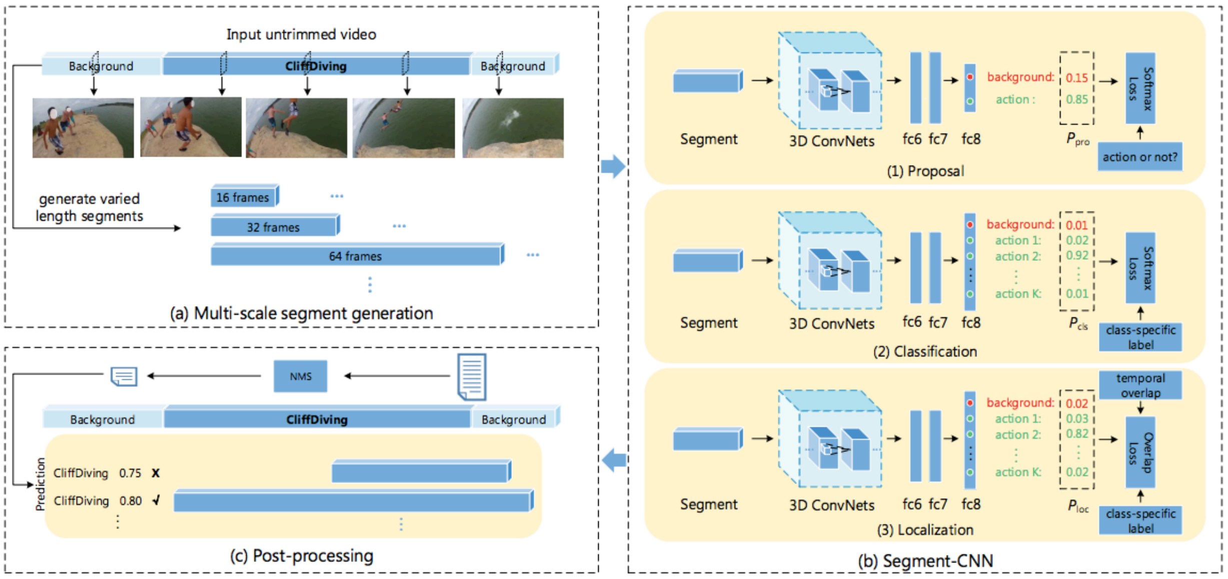

Figure 2. SCNN architecture.

To improve temporal localization accuracy, recently a method called TURN [5] proposed to use temporal boundary regression. TURN [5] takes sliding windows and their surrounding context as input and refines their temporal boundaries by learning a boundary regressor. In the following section, we introduce TURN architecture in detail.

B. *TURN Architecture*

The architecture of TURN is shown in Figure 3, which includes three modules: video unit processing, clip pyramid modeling, proposal prediction. In the following sections, we will introduce the three modules and the loss function.

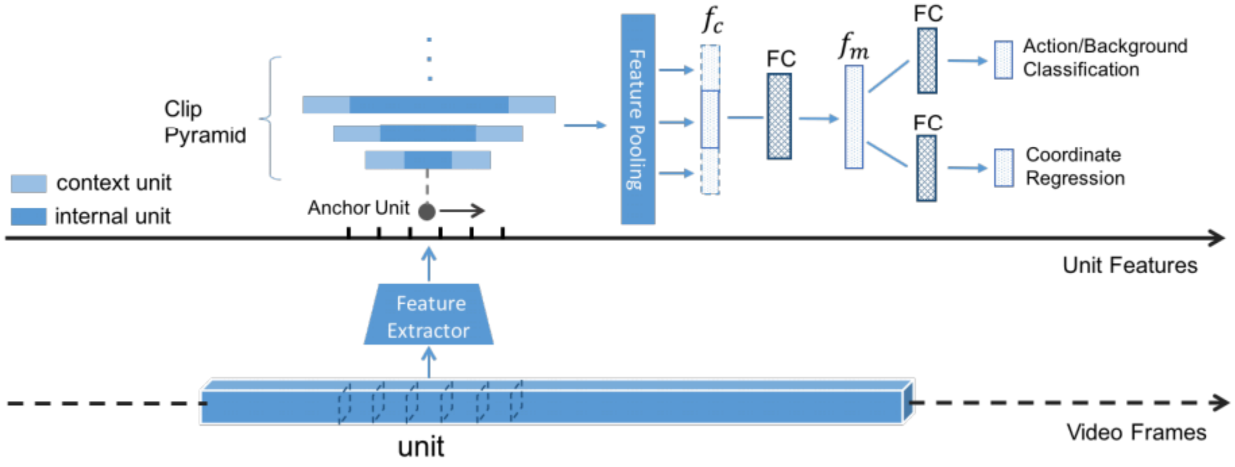

Figure 3. TURN architecture.

Video unit processing

A video V contains T frames, $V = \{t_i\}$, and is divided into T/n_u consecutive video units , where nu is the frame number of a unit. A unit is represented as $u = \{t_i\}_{s_f}^{n_u+s_f}$, where s_f is the starting frame, $s_f + n_u$ is the ending frame.

Each unit is processed by a visual encoder Ev to get a unit-level representation fu = Ev(u). In our experiments, C3D [8], optical flow based CNN model [9] and RGB image CNN model [6] are investigated.

Clip Pyramid Modeling

A clip (i.e. window) c is composed of units, $c = \{u_j\}_{s_u}^{s_u+n_c}$, where s_u is the index of starting unit and n_c is the number of units inside c. $e_u = s_u + n_c$ is the index of ending unit, and $\{u_j\}_{s_u}^{e_u}$ is called internal units of c. Besides the internal units, context units for c are also modeled. $\{u_j\}_{s_u-n_{ctx}}^{s_u}$ and $\{uj\}_{e_u}^{e_u+n_{ctx}}$ are the context before and after c respectively, n_{ctx} is the number of units we consider for context. Internal feature and context feature are pooled from unit features separately by a function P. The final feature fc for a clip is the concatenation of context features and the internal features; fc is given by

$$f_c = P(\{u_j\}_{s_u-n_{ctx}}^{s_u}) \,||\, P(\{u_j\}_{s_u}^{e_u}) \,||\, P(\{u_j\}_{e_u}^{e_u+n_{ctx}})$$

where || means vector concatenation. The original TURN paper uses average pooling for P . Besides average pooling, we also tried max pooling. As we think average pooling only captures the overall semantic meaning of the clip, however, it may ignore the most representative part of the video, which may have significant effect on the proposal prediction.

Unit-level Temporal Boundary Regression

we design a unit regression model that takes a clip-level representation fc as input, and have two sibling output layers. The first one outputs a confidence score indicating whether the input clip is an action instance. The second one outputs temporal coordinate regression offsets. The regression offsets are

$$o_s = s_{clip} - s_{gt}, \qquad o_e = e_{clip} - e_{gt}$$

Loss Function

We use a multi-task loss L to jointly train classification and coordinates regression

$$L = L_{reg} + L_{cls}$$

$$L_{reg} = \frac{1}{N}\sum_{i=1}^{N} l_i^*[|o_{s,i} - o_{s,i}^*| + |o_{e,i} - o_{e,i}^*|]$$

L1 distance is adopted. l_i^* is the label, 1 for positive samples and 0 for background samples. N_{pos} is the number of positive samples. The regression loss is calculated only for positive samples.

C. *Metrics*

AR-F curve. This metric is for action proposal evaluation. We measure average recall as a function of proposal frequency (F), which denotes the number of retrieved proposals per second for a video.

mAP @ IoU=x, this metric is for action detection evaluation. A prediction is marked as correct only when it has the correct category prediction, and has Intersection Over Union (IoU) with ground truth instance larger than the overlap threshold. We use threshold at 0.1, 0.2, 0.3, 0.4, 0.5 for evaluation. Note that redundant detections are not allowed.

III. EXPERIMENT SETTINGS

A. *AWS Settings*

In AWS, we use EC2 service for model training. When launching EC2 instance, we choose Ubuntu Server 16.04 LTS (HVM), SSD Volume Type as AMI. Amazon EC2 provides a wide selection of instance types optimized for different use cases. Instances are virtual server where we can run our applications. The instance type we choose is GPU compute p2.xlarge which contains 4 virtual CPUs, 61 GiB memory. GPU compute instance provides general-purpose GPUs along with high CPU performance, large memory and high network speed for application requiring massive floating point process power. Thus, it is suitable for machine learning problems.

B. *Environment Configuration*

We installed TensorFlow 0.12, Cuda Toolkit 8.0 and cuDNN v5.0 at our EC2 instance. TensorFlow is a deep learning library open-sourced by Google. TensorFlow provides primitives for defining functions on tensors and automatically computing their derivatives. TensorFlow not only a pre-defined library of algorithms, but also a low-level library where users can orchestrate and design complex algorithms and provide large flexibility.

CUDA and cuDNN is indispensable for TensorFlow with GPU support. CUDA, as know as Compute Unified Device Architecture, is a parallel computing platform and programming model from NVIDIA. With CUDA Toolkit, we can develop, optimize and deploy our applications on GPU-accelerated cloud-based platforms. cuDNN, as known as CUDA Deep Neural Network library, is a GPU-accelerated library of primitives for deep neural networks. cuDNN provides highly tuned implementations for standard routines such as forward and backward convolution, pooling, normalization, and activation layers. With cuDNN, we can focus more on training neural networks and developing software applications and do not need to worry about low-level GPU performance tuning.

C. *Dataset*

The dataset we use is THUMOS 14. The THUMOS 14 challenge introduced thousands of untrimmed videos in validation, background and test sets for 101 action classes providing the community with the first-of-its-kind dataset for action recognition and temporal detection in realistic settings with a standardized evaluation protocol [8].
The temporal action localization part of THUMOS-14 contains over 20 hours of videos from 20 sports classes. This part consists of 200 videos in validation set and 213 videos in test set. TURN model is trained on the validation set, as the training set of THUMOS-14 contains only trimmed videos.

D. Model Training

For model training, we follow the original TURN paper, set the context number as 4, unit size as 16 or 32, and train the model for 10 epochs. We use Adam optimizer to train the model, and set learning rate as 0.001. The model is trained on a Nvidia K80 GPU. Totally, it takes about 2 hours to make the model converge.

III. RESULTS ANALYSIS

A. Training Loss virtualization

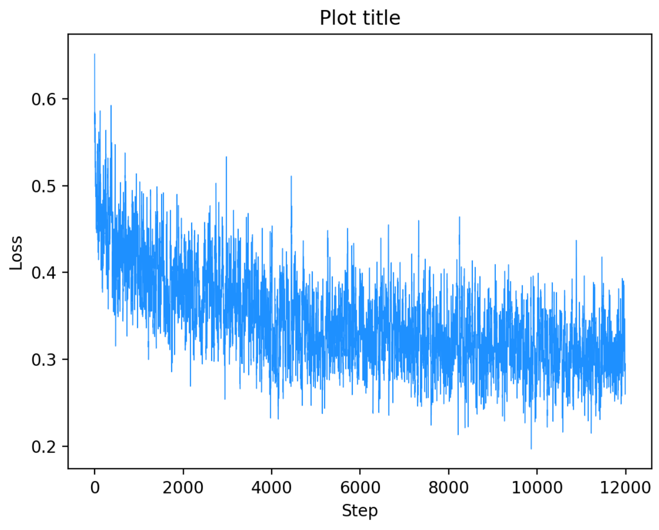

Figure 4. Training Loss of Mean-Pooling

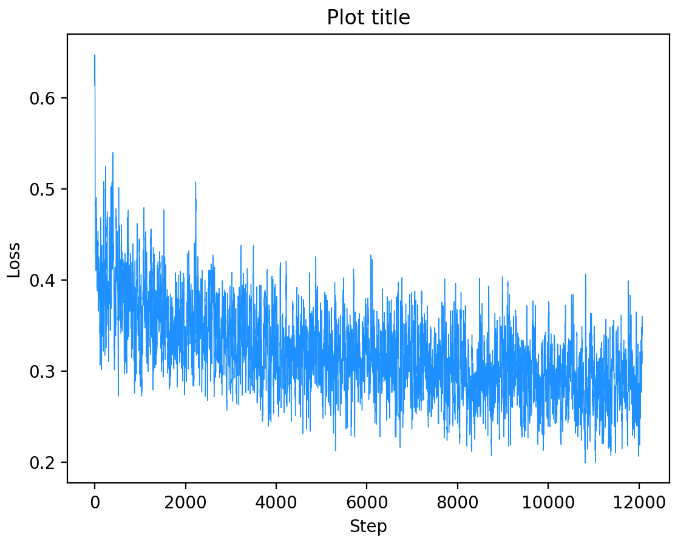

Figure 4. Training Loss of Max-Pooling

In Fig 4. we plot the loss of the model with mean-pooling and max-pooling from step 0 to step 12000 of training. In training loss virtualization, we applied Local mean filtering, which calculate the average loss of ten nearby steps as the loss of current step. Such a process filters out the noise and make the visualization of the loss more consistent. From the figure, we can see that, the value of the loss consistently decreases, which shows the effectiveness of the training.

B. Evaluation on Unit-level Features

In this section, we follow the original TURN paper and evaluate TURN methods with different system settings and compare the results. We test TURN with three unit-level features to assess the effect of visual features on AR performance: C3D [9] features, RGB CNN features with temporal mean pooling and dense flow CNN [1] features. The results are shown in Figure 5.

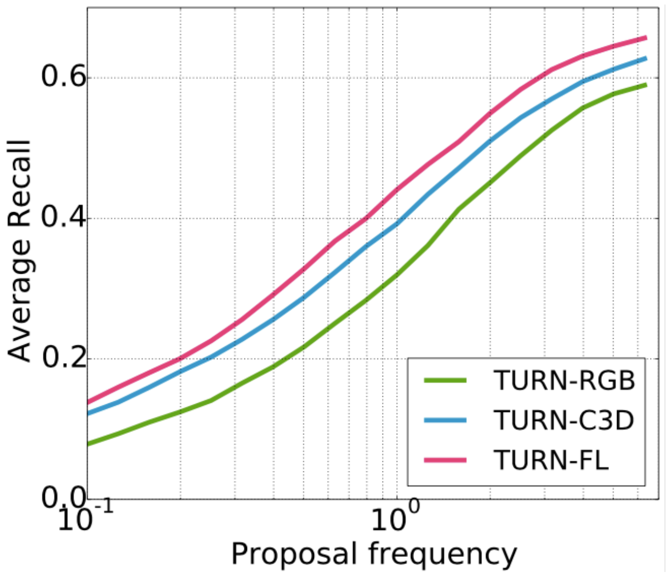

Figure 5. TURN methods with different system settings

We can see that Flow feature works best with TURN for action proposal generation, which is the same as the resutls in original TURN paper [5], dense flow CNN feature (TURN-FL) gives the best results, indicating optical flow can capture temporal action information effectively.In contrast, RGB CNN features (TURN-RGB) show inferior performance and C3D (TURN-C3D) gives competitive performance.

C. Evaluate Max-pooling and Mean-pooling

In this part, we evaluate the performance of different methods on clip-level feature construction. The original TURN uses average pooling method to construct clip-level features, however, we think that average pooling only captures the overall semantic meaning of the clip, however, it may ignore the most representative part of the video, which may have significant effect on the proposal prediction. In the following Table, we show the results on max-pooling and mean-pooling.

TABLE I
COMPARISON OF MEAN-POOLING AND MAX-POOLING

Method	AR @ F=1.0(%)
Mean-Pooling	38.7
Max-Pooling	38.3

We can see that, max-pooling slightly outperforms the mean-pooling, which shows the effectiveness of our proposed max pooling method. We believe the reason is that max-pooling better at capturing signiture features in a clip, which helps the proposal prediction.

D. Compare TURN with state-of-the-art method

We compare TURN with the state-of-the-art methods under AR-F, Recall@F-tIoU metrics. The TAP generation methods include DAPs [4], SCNN-prop [3], Sparse- prop [2], sliding window, and random proposals. For DAPs, Sparse-prop and SCNN-prop, we plot the curves using the proposal results provided by the authors. For TURN, we train and test the model on Amazon EC2.

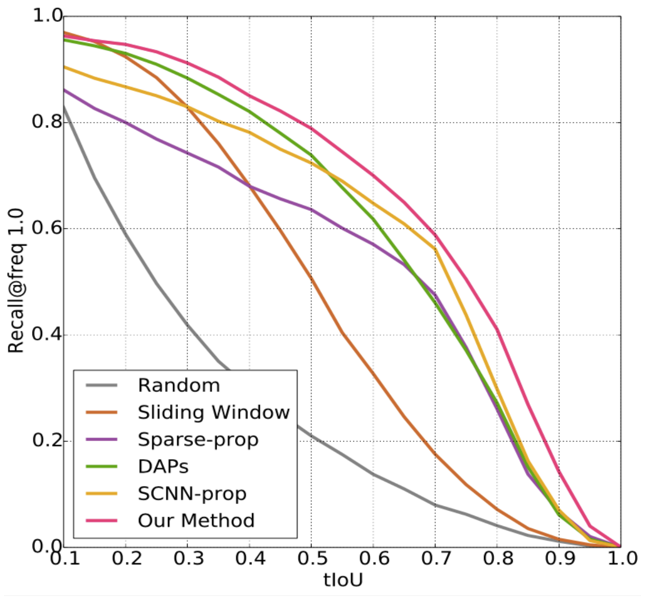

Figure 6. recall rate at different tIoU.

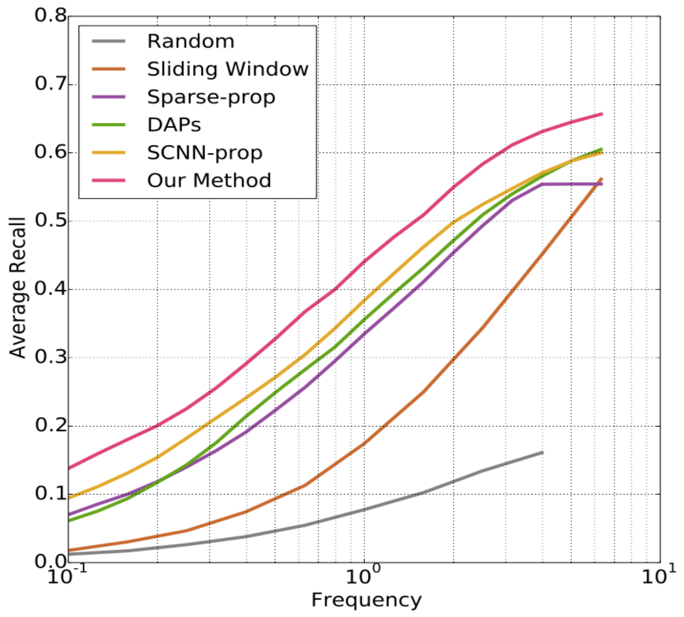

Figure 7. Recall rate at different frequency.

We can see that, TURN outperforms other methods by a large margin, which shows the correcetness of our implementation and training.

TABLE 2
COMPARISON OF AR AND RUN-TIME

Method	AR @ F=1.0(%)	FPS
DAPs [10]	35.7	134.3
SCNN-prop [3]	38.3	60.0
Sparse-prop [11]	33.3	10.2
TURN-FL-16	43.5	129.4
TURN-FL-32	42.4	260.6
TURN-C3D-16	39.3	880.8

Table 2 reveals the recall (AR@F=1.0) and run time (FPS) performance of DAPs, SCNN-prop, Sparse-prop and three TURN variants. The difference between three TURN variants is that the number of unit size is 16 or 32. From Table, by comparing TURN-FL-16 with TURN-FL-32, we can see that a smaller unit size has higher recall rate and higher computational complexity. This is because, smaller unit size leads to an increase number of unit size and a increase temporal coordinate precision.

By comparing between the three TURN variants, the TURN-C3D-16 is way faster than the TURN-FL-16, however, the TURN-FL-16 has higher performance. By comparing TURN variants with other state-of-art methods, TURN-FL-16 has higher AR performance and competitive run-time performance. TURN-FL-32 has higher AF performance and higher run-time performance. TURN-C3D-16 has the highest run-time performance and also has higher AF performance than all the state-of-art methods. In conclusion, TURN model has a better overall performance than the state-of-art methods.

TABLE 3
COMPARISON OF AR AND RUN-TIME

tIoU	0.1	0.2	0.3	0.4	0.5
Yeung et al. [12]	48.9	44.0	36.0	26.4	17.1
Yuan et al. [13]	51.4	42.6	33.6	26.1	18.8
S-CNN [3]	47.7	43.5	36.3	28.7	19.0
TURN-C3D-16 + SVM	46.4	41.5	34.3	24.9	16.4
TURN-FL-16 +SVM	48.3	43.2	35.1	26.2	17.8
TURN-C3D-16 + CNN	48.8	45.5	40.3	31.5	22.5
TURN-FL-16 + S-CNN	54.0	50.9	44.1	34.9	25.6
TURN-FL-16-**Maxpool** + S-CNN	54.3	51.3	44.7	35.2	26.2

The Table 3 shows temporal action localization performance of state-of-art methods and TURN models under different tIoU thresholds. Under same tIoU threshold, we can see that by applying SVM, TURN can have a competitive performance. In addition, by applying S-CNN localizer, TURN can have a better performance than all the start-of-art methods.

TURN can help with action localization. On the one hand, TURN help us eliminate the influence of unimportant component, on the other hand, TURN can increase the performance of action localization.

We also test the max-pooling method, which is originally proposed by us, as shown in the last row. We can see that max-pooling gives a further improvement over TURN, and achieve state-of-the-art method. The results prove the effectiveness of our proposed max-pooling method. We believe the reason is that max-pooling method bettering at capturing signature and important information inside the clip, which benefit the localization process and action detection.

IV. CONCLUSIONS

In this paper, we tackle the problem of temporal action detection and action proposal generation, which have great impact on video analysis and are important problems in computer vision fields. We train and test state-of-the-art method TURN with different system settings. To further improve the performance, we propose to use max-pooling to replace the mean-pooling in clip-level feature construction, as max-pooling may help to capture signature information and feature inside a clip, which may benefit the localization process. We evaluate the original TURN model and the max-pooling modification on THUMOS-14 dataset. For computing platform, AWS EC2 instance (p2.xlarge) is used, which contains a NVidia K80 GPU and 4 vCPU with 61 GB memory. The experimental results show the correctness and effectiveness of our model training and test of TURN. By applying max-pooling method, we further boost the performance on THUMOS-14.

REFERENCES

[1] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for ac- tion recognition in videos. In NIPS, 2014.

[2] Bharat Singh, Tim K. Marks, Michael Jones, Oncel Tuzel, and Ming Shao. A multi- stream bi-directional recurrent neural network for fine-grained action detection. In CVPR, 2016.

[3] Zheng Shou, Dongang Wang, and Shih-Fu Chang. Temporal action localization in untrimmed videos via multi-stage cnns. In CVPR, 2016.

[4] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. In ICLR, 2014.
[5] Jiyang Gao, Zhenheng Yang, Chen Sun, Kan Chen, and Ram Nevatia. Turn tap: Temporal unit regression network for temporal action proposals. In ICCV, 2017.
[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In CVPR, 2016.
[7] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In ICML, 2015.
[8] http://crcv.ucf.edu/THUMOS14/home.html
[9] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In ICCV, 2015.
[10] V. Escorcia, F. C. Heilbron, J. C. Niebles, and B. Ghanem. Daps: Deep action proposals for action understanding. In *ECCV*, 2016.
[11] F. Caba Heilbron, J. Carlos Niebles, and B. Ghanem. Fast temporal activity proposals for efficient detection of human actions in untrimmed videos. In *CVPR*, 2016.
[12] S. Yeung, O. Russakovsky, G. Mori, and L. Fei-Fei. End- to-end learning of action detection from frame glimpses in videos. In *CVPR*, 2016.
[13] J. Yuan, B. Ni, X. Yang, and A. A. Kassim. Temporal action localization with pyramid of score distribution features. In *CVPR*, 2016.